Mr. Mirko Čubrilo
Fakultet organizacije i informatike
Varaždin

UDK: 007.52
Znanstveni rad

NEKI ASPEKTI MODELIRANJA LJUSKI EKSPERTNIH SISTEMA

Temu ovoga rada čini razmatranje nekih aspekata modeliranja ljuski ekspertnih sistema. To se prvenstveno odnosi na formalne metode specifikacije pojedinih modula ljuske. Vezano za pretpostavljene problemske sredine, važna je specifikacija grafičkog editora formula jezika računa predikata prvog reda. U radu je dana BN - specifikacija gramatike jezika računa predikata prvog reda kao osnova za generiranje sintaktičkog analizatora. Generirani sintaktički analizator služi dalje kao predprocesor deduktivnog mehanizma. Nadalje, analiziran je primjer SF-specifikacije fragmenta hipotetičkog editora i demonstrirana mogućnost implementacije njegovih funkcija u jeziku PDC Prolog.

Ekspertni sistem; ljuska ekspertnih sistema; deduktivni mehanizam; formalna specifikacija; BN-specifikacija; SF-specifikacija; prolog.

1. Radna sredina

Programske paradigme

Dvije su moderne programske paradigme. Jedna obuhvaća proceduralno programiranje, a druga tzv. programiranje za umjetnu inteligenciju (artificial intelligence programming). Teorijsku osnovu za prvu čini teorija algoritama, a za drugu mnogobrojni logički računi (račun sudova, račun predikata prvog reda, modalni računi [sudova i predikata], računi funkcijskih ovisnosti, višeznačnih ovisnosti, ovisnosti spajanja, vezani za relacijski model podataka, itd).

Za programe pisane u proceduralnim jezicima programiranja karakteristično je implicitno specificiranje onoga što treba uraditi i eksplicitno i detaljno specificiranje kako isto uraditi.

U programima pisanim u neproceduralnim jezicima programiranja naglasak je na opisu problema (što uraditi), dok je pitanje kako to uraditi prepušteno sistemu i za korisnika je transparentno. Za ovaj tip programa karakteristično je da rješavaju složene probleme za čije rješenje ne postoje očigledni algoritmi i da oni za svoje rješavanje zahtijevaju primjenu (preko implementacije u odgovarajući jezik, odnosno program) simboličkog zaključivanja.

Jedan segment u domeni programiranja za umjetnu inteligenciju zauzimaju ekspertni sistemi.

Ekspertni sistemi

Ekspertni sistemi slični su drugim programima iz ove domene po tome što koriste:

- intenzivno pretraživanje prostora stanja problema
- heurističke metode pretraživanja
- automatsko rezoniranje (zaključivanje)
- odvajanje znanja o problemu od upravljanja procedurom dostizanja rješenja
- korisniku orijentirana sučelja.

Ekspertni sistemi razlikuju se od drugih programa iz ove domene po:

- svrsi
- komunikaciji s korisnikom.

Svrha im je da probleme iz pojedinih uskih problemskih oblasti (pojedine grane medicinske dijagnostike, tehnička dijagnostika, kemijska organska sinteza, konceptualna klasifikacija, logičko-aritmetički i logičko-kombinatorni problemi,.) rješavaju snagom eksperata.

Komunikacija s korisnikom je interaktivna sa sučeljem koje podržava ograničeno komuniciranje na prirodnom jeziku.

Komponente ekspertnog sistema

Glavne komponente svakog ekspertnog sistema (konceptualno gledane) jesu:

- baza znanja
- mehanizam zaključivanja
- korisničko sučelje

Na nivou ekspertnog sistema kao programa, nabrojenim komponentama odgovaraju pripadni programski moduli u okviru glavnog programa.

Ljuske ekspertnih sistema

Za ljusku ekspertnih sistema u odnosu na pojedinačni ekspertni sistem karakteristična je izmjenjivost modula za rukovanje znanjima, u manjoj mjeri i mehanizma zaključivanja i konstantno (neizmjenjivo) korisničko sučelje.

Cilj ovoga rada je da oblikuje (konceptualno i organizacijski) jedan model ljuske ekspertih sistema za rješavanje problema iz nekoliko specifičnih problemskih oblasti te da analizira mogućnosti njegove realizacije sredstvima jezika PDC Prolog. Posebna pažnja bit će posvećena analizi svojstava i mogućnostima realizacije (grafičkog) editora formula računa predikata prvog reda kao jednog od modula u okviru kojeg se problemi rečenog tipa mogu prirodno formulirati.

2. Problemske oblasti

Među mogućim problemskim oblastima posebno se osvrćem na slijedeće:

1. logičko aritmetički problemi
2. planiranje (robotskih) akcija
3. gramatike oblika
4. relacijske baze podataka (pitanje izvedivosti višeznačnih ovisnosti)

Svaka od predloženih problemskih oblasti bit će ilustrirana s jednim do dva problema. Svrha detaljne prezentacije problema je u tome da omogući detaljniju specifikaciju strukture pojedinih modula ljuske ekspertnih sistema.

2.1. Logičko - aritmetički problemi

PROBLEM 1: ([Bizam & Herczeg, 1972])

Dano je pet kutija u različitim bojama (bijela, crna, crvena, plava i zelena) i po dvije kuglice svake od navedenih boja. Poznato je slijedeće:

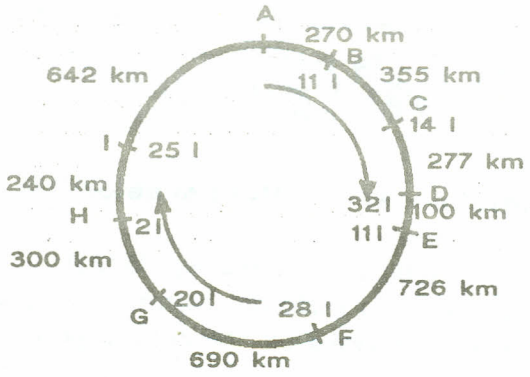
0. U svakoj kutiji su po dvije kuglice.
1. Ni jedna kuglica ne leži u kutiji iste boje.
2. U crvenoj kutiji nema plavih kuglica.
3. U kutiji neutralne boje leži po jedna crvena i zelena kuglica.
4. U crnoj kutiji leže kuglice hladnih boja.
5. U jednoj od kutija leži jedna bijela i jedna plava kuglica.
6. U plavoj kutiji nalazi se jedna crna kuglica.

Nadalje, u formulaciji problema objašnjava se da su neutralne crna i bijela boja te da su hladne boje zelena i plava. Sam problem sastoji se u tome da se odredi raspored kuglica po kutijama konzistentan s postavljenim uvjetima.

Formalna reprezentacija ovog problema bit će dana u odjeljku 3.2.

PROBLEM 2: ([Bizam & Herczeg, 1972])

Na kružnoj automobilskoj pisti postavljene su benzinske stanice (pumpe) A — I, tako da je u svakoj od njih pospremljena određena količina benzina. Nadalje, poznata je udaljenost između svake dvije susjedne stanice. Situacija je opisana crtežom na narednoj strani. Automobil troši 4 l benzina na svakih 100 km prevaljenog puta. Problem je u slijedećem : odrediti bar jednu (ako postoji) benzinsku stanicu uz uvjet da polazeći iz nje automobil obiđe čitavu pistu i vrati se na polaznu točku. Pretpostavka je da se rezervoar za gorivo puni tek na polaznoj stanici i da je prije toga bio prazan.

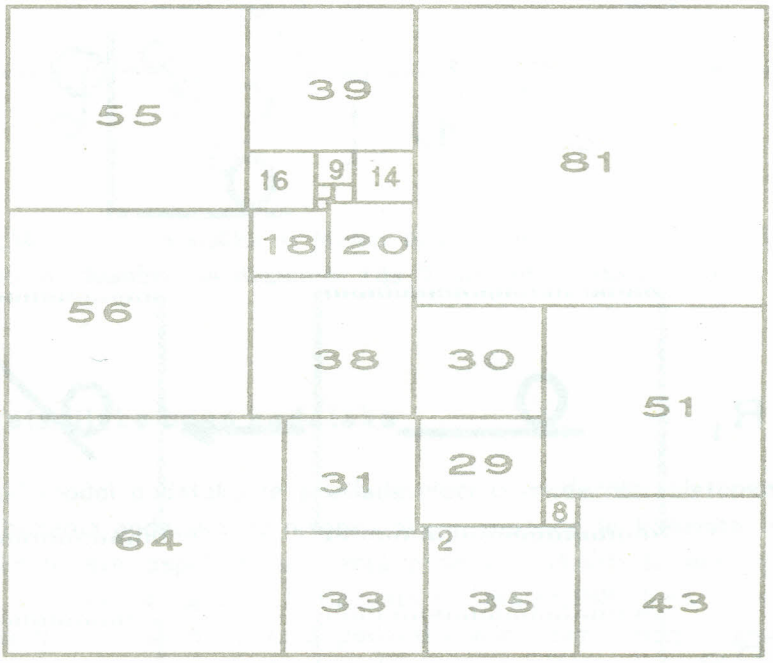


2.2. Planiranje robotskih akcija

Poznato je ([Jaglom, 1968]) da se svaki kvadrat može razbiti na određeni broj drugih kvadrata. Primjer jednog takvog netrivijalnog rastava (kvadrata s mjernim brojem duljine stranice jednakim 175) dan je na crtežu.

PROBLEM: Od kvadrata koji čine rastav složiti polazni kvadrat.

Problem očito spada u probleme planiranja (redoslijeda) akcija u odgovarajućoj, dobro definiranoj radnoj sredini.



2.3. Gramatike oblika ([Stiny, 1975])

Oblik: skup dužina u ravnini dobiven nekom slikovnom specifikacijom.

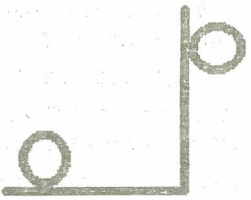
O^+: zatvarač skupa oblika O, zatvoren s obzirom na operaciju unije oblika i euklidske transformacije (translacija, rotacija, sličnost)

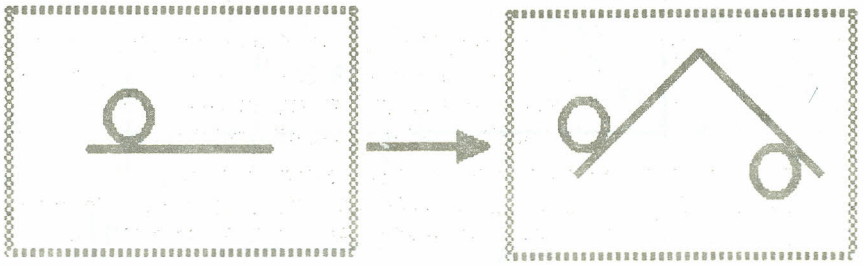
$O^{\uparrow} = O^{+} \cup \Phi$ (Φ je "prazan oblik")

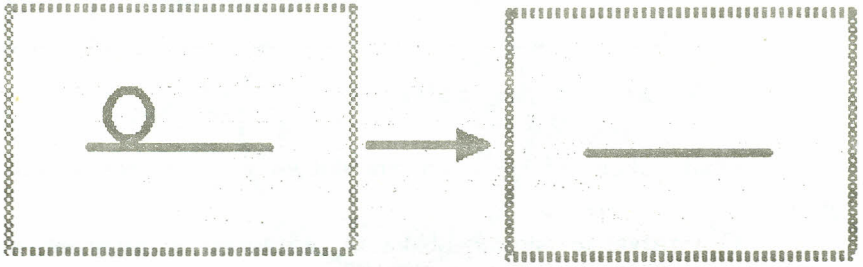
Gramatika oblika (shape grammar)

SG $= \langle V_T, V_M, R, I \rangle$, gdje je V_T skup terminalnih objekata, V_M skup neterminalnih objekata, R skup pravila izvoda (zamjene objekata) i I jednočlani skup koji sadrži inicijalni objekt. Slijedi primjer gramatike oblika prema navedenoj referenci.

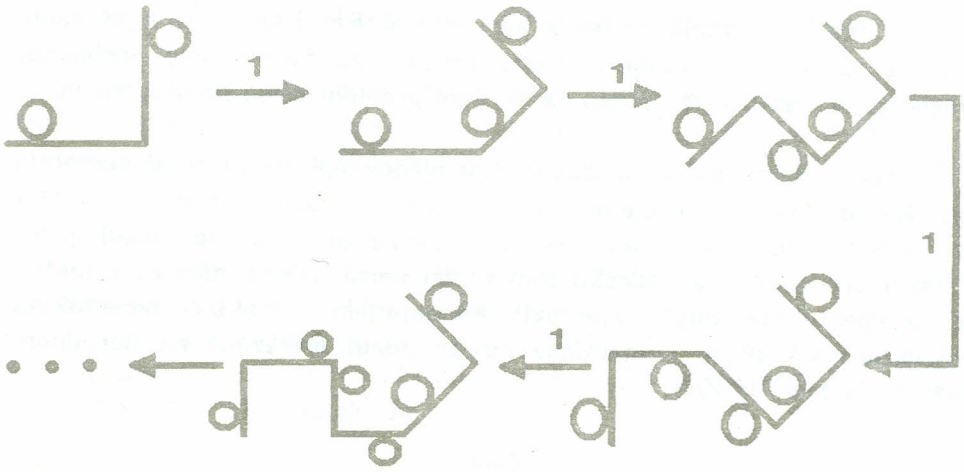
SG1 : $V_T = \{$ —— $\}$, $V_M = \{ O \}$

$I = \{$  $\}$

R_1 

R_2 

Primjer izvoda u SG1



PROBLEM 1: Za dva oblika u jeziku nad istom gramatikom oblika ispitati da li su međusobno ekvivalentni (da li se jedan može svesti na drugoga)?

2.4. Relacijske baze podataka

Relacijski model podataka je prevladavajući u modernim sistemima za upravljanje bazama podataka. U njegovu okviru moguće je konzistentno formulirati i riješiti sve aspekte rukovanja podacima (kontrola pristupa, konzistentnost transakcija, upiti, ...). Poznato je [Maier,1983] da se semantički odnosi u relacijskim bazama podataka vrlo dobro mogu opisati semantikom računa funkcijskih i višeznačnih ovisnosti.

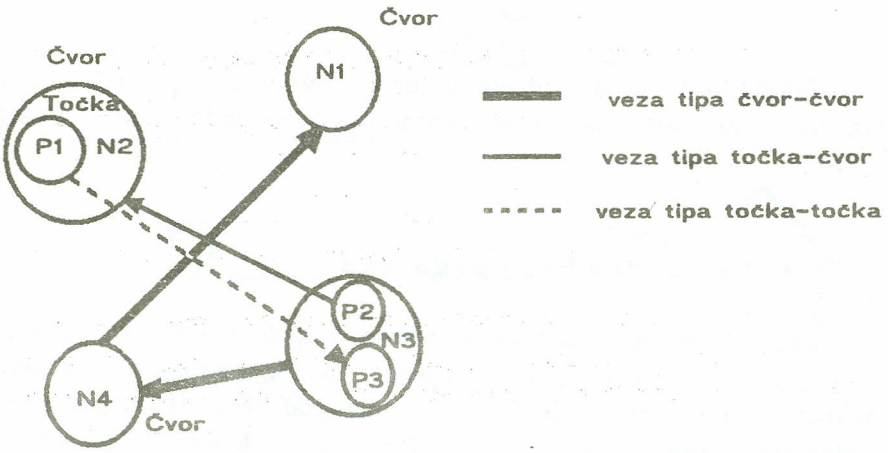
PROBLEM: Za dani skup (višeznačnih) ovisnosti V i ovisnost $X \twoheadrightarrow Y$ ispitati da li je ovisnost $X \twoheadrightarrow Y$ sintaktička posljedica skupa ovisnosti V?

3. Reprezentacija problema

3.1. Razine reprezentacije problema

Ljuska ekspertnih sistema kao interaktivni sistem za rješavanje problema mora imati razvijenu radnu sredinu za formuliranje problema. Poželjno je, a često i potrebno, da se dani problem opiše na više razina.

Jednu razinu opisa problema čini njegov opis u svrhu dokumentiranja. Na toj razini razlikujemo samo tekstualni i samo slikovni (grafički) opis, odnosno njihovu kombinaciju. Za realizaciju (implementaciju) programskih sredstava za podršku ovoj razini opisa najviše odgovara metodologija hiperteksta (engl. hypertext). Konceptualno, struktura hiperteksta je struktura višeslojne semantičke mreže poput prikazane na narednom crtežu ([Hashim, 1990]).



Do sada često rabljeni apstraktni (formalni) model strukture hiperteksta formuliran je u članku [Campbell & Goodman, 1988].

Drugu razinu opisa (reprezentacije) problema čini opis problema u svrhu njegovog rješavanja. Drugim riječima, radi se o formalizaciji problema u okviru adekvatnog formalnog jezika (poput jezika računa sudova, jezika računa predikata prvog reda, jezika fragmenata teorije skupova, itd.). Rezultat formalizacije često je skup formula u dotičnom formalnom

jeziku. Takav skup formula predstavlja onda ulaz za deduktivni mehanizam u okviru ljuske ekspertnih sistema koji zatim pronalazi rješenje ako ono uopće postoji.

3.2. Primjer formalne reprezentacije problema

Ilustracije radi, dajem formalizaciju problema vezanog za raspored kuglica po kutijama (problem br. 1 iz točke 2.1) u okviru jezika računa predikata prvog reda s jednakošću kao "sistemskim" (built in) predikatom.

Konstante u jeziku za boje kuglica i kutija su c, b, p, cr i z redom za crnu, bijelu, plavu, crvenu i zelenu boju.

Koristimo logičke veznike "$\neg$" ("ne"), "$\vee$" (" ili "), "$\wedge$" (" i "), "$\rightarrow$ " ("povlači") i " $\leftrightarrow$ " ("ekvivalentno") te kvantifikatore " $\forall$ " (" za svaki ") i " $\exists$ " (" postoji ").

Neka je $K(x,y,z)$ predikat koji je u svakoj interpretaciji istinit ako, i samo ako u kutiji boje x leže kuglice boja y i z. Iz tekstualnog opisa problema nastaje naredni skup formula u rečenom jeziku.

F0: $\forall x \exists y \exists z K(x,y,z)$

F1: $\forall x \forall y \forall z (K(x,y,z) \rightarrow (\neg(y = x) \wedge \neg(z = x)))$

F2: $\forall y \forall z (\neg K(cr, p, z) \wedge \neg K(cr, y, p))$

F3: $K(b, cr, z) \vee K(c, cr, z)$

F4: $K(c,p,p) \vee K(c,p,z) \vee K(c,z,z)$

F5: $\exists x K(x,b,p)$

F6: $\exists z K(p,c,z)$

Gornjim formulama mora se pridodati formula kojom se kazuje da je predikat K simetričan s obzirom na drugi i treći argument:

F7: $\forall x \forall y \forall z (K(x,y,z) \rightarrow K(x,z,y))$,

kako bi se u nekima od njih izbjeglo gomilanje atomarnih formula.

Posljednja formula predstavlja primjer metaznanja o problemskoj oblasti. Zbog osiguranja jednoznačnosti čitanja (interpretacije) formula potrebno je u njihovom zapisivanju koristiti zagrade na dosljedan i konzistentan način. Uobičajeno je u zapisima formula, poput prethodnih, izostavljati pojedine parove zagrada ako to ne smeta jednoznačnosti čitanja formule. Ako bismo u posljednjoj formuli dosljedno pisali zagrade, onda bi zapis izgledao ovako:

$$F7' : (\forall x(\forall y(\forall z(K(x,y,z) \rightarrow K(x,z,y)))))$$

O deduktivnom mehanizmu ugrađenom u ljusku ekspertnih sistema ovisi dalje potreba za eventualnim dodatnim transformacijama dobivenog skupa formula. Ako bi to, na primjer, bio klasični sistem zasnovan na pravilu rezolucije za račun predikata prvog reda (vidjeti naprimjer [Čubrilo, 1989]), onda bi polazni skup formula trebalo transformirati u njemu (bar) ekvikontradiktoran skup disjunkta. To bi ovdje bio skup:

$$D = \{ K(x,f(x),g(x)),\ \neg K(x,y,z) \vee \neg(y = x),\ \neg K(x,y,z) \vee \neg(z = x),$$
$$\neg K(a,p,z),\ \neg K(a,y,p),\ K(b,a,z) \vee K(c,a,z),\ K(c,p,p) \vee K(c,p,z) \vee$$
$$\vee K(c,z,z),\ K(c1,b,p),\ K(p,c,c2),\ \neg K(x,y,z) \vee K(x,z,y) \},$$

gdje su c1 i c2, odnosno f i g konstante, odnosno funkcije, nastale eliminacijom kvantifikatora iz formula F0 – F7. Neka deduktivna pravila, zasnovana također na pravilu rezolucije, poput onog opisanog u članku [Bhatta & Karnick, 1991] ne zahtijevaju eliminaciju kvantifikatora iz formula.

4. Metode specifikacije

4.1. Konceptualna specifikacija svojstava (grafičkog) editora formula računa predikata

Metode specifikacije modula u okviru ljuske ekspertnih sistema po svojoj namjeni moraju omogućiti formalnu specifikaciju istih jer formalna specifikacija predstavlja "samo korak" do implementacije. Kada se govori o konceptualnoj specifikaciji, onda se prvenstveno misli na "popis želja" koje se odnose na specificirani modul. U tom smislu u ovom se odjeljku govori o konceptualnoj speciifikaciji (grafičkog) editora formula jezika računa predikata prvog reda.

Kako je račun predikata prvog reda prirodna sredina za formalizaciju problema iz navedenih problemskih oblasti, nužno je da ljuska ekspertnih sistema za rješavanje takvih problema u svom sastavu (kao modul) ima (grafički) editor formula jezika računa predikata.

Zadatak je takvog editora da omogući zapis i editiranje formula računa predikata u uobičajenoj logičkoj notaciji. Govorimo o grafičkom editoru jer klasični editori teksta ne podržavaju rad sa specijalnim znakovima iz abecede jezika računa predikata. Zbog toga je potrebne znakove nužno kreirati nekim grafičkim programom i (ili) ih "skinuti"s ekrana odgovarajućim programom za kopiranje ekrana, te ih kao binarne blokove organizirati u jedan ili više setova znakova i kao takve ih koristiti.

Klasični skup naredbi za operacije na tekstu i s tekstom ovdje može biti značajno reduciran. Na primjer, malo je vjerojatno da će duljina bilo koje formule premašiti 70 znakova, što je uobičajena mimimalna duljina retka na ekranu. Zbog toga nema potrebe za operacijom preloma retka. Isto tako nema potrebe za operacijama izravnavanja retka, izravnavanja paragrafa, itd.

Nužne naredbe na uobičajeni način dijelimo na grupe sličnih, i to:

a) naredbe za upravljanje značkom (kursorom)

U ovu skupinu naredbi spadaju naredbe za pomicanje pokazivača na redak iznad ili ispod retka u kojem se trenutno nalazi, naredbe za pomicanje za jedno mjesto (znak) lijevo ili desno od trenutne pozicije u retku, naredbe za pomicanje na početak lijeve ili desne riječi u odnosu na riječ u retku na kojoj je značka pozicionirana, itd.

b) naredbe za umetanje i brisanje

Ovamo ubrajamo naredbe za umetanje praznog retka, brisanje znaka, brisanje riječi, brisanje retka do njegova početka ili kraja s obzirom na trenutnu poziciju, itd.

c) naredbe za traženje i zamjenu

Ovamo spadaju naredbe za traženje zadane riječi u tekstu i (uzastopnu) zamjenu te riječi nekom drugom zadanom riječi.

d) naredbe za rukovanje datotekama

Ovdje ubrajamo naredbe za učitavanje i pospremanje datoteka, otvaranje novih datoteka, preimenovanje datoteka, pretraživanje kataloga datoteka (direktorija), itd.

Zbog potreba sintaktičkog analizatora (parsera) poželjno je imati specijalne znakove za početak i kraj formule.

4.2. Karakteristike formalnih metoda specifikacije

"Opći dojam" programskog paketa u mnogome ovisi o kvaliteti njegovog korisničkog sučelja. Dok su korisnici programskih paketa bili mahom specijalizirani stručnjaci, pitanjima razvoja kvalitetnog korisničkog sučelja nije poklanjana prevelika pažnja, kako od strane proizvođača, tako i od strane korisnika. Prodorom informatičke tehnologije u gotovo sva područja proizvodnje, znanosti, kulture i obrazovanja, a time i proširenjem kruga korisnika računala i pripadnih aplikativnih programa, pitanje dizajna kvalitetnog korisničkog sučelja postaje jedno od najvažnijih. Da bi se moglo odgovoriti rastućim zahtjevima s obzirom na raznolikost i rastuću složenost korisničkih sučelja, razvijen je čitav niz metodologija specifikacije.

Potreba za sredstvima formalne specifikacije proizlazi i iz sve veće složenosti aplikativnih programa, koja je opet posljedica rastuće složenosti problema koje oni rješavaju.

Među poznatijima su specifikacija na osnovi teorije komunikacije sekvencijalnih procesa (CSP specifikacija) [Alexander,1990], specifikacija sredstvima gramatika "zadatak-akcija" (task-action grammars), Z specifikacija, BN-specifikacija i SF-specifikacija (Set Function specification) [Sufrin, 1982] i [Berztiss, 1990].

Formalne metode specifikacije moraju ispunjavati čitav niz prirodnih zahtjeva, poput nekoliko narednih:

- mora omogućiti formalnu definiciju odgovarajućeg modula (korisničkog sučelja, editora, ...)

- mora biti potpuna u smislu da obuhvati cjelinu interakcije između korisnika i programa (korisnički ulaz, odgovor sistema i veza s izvršnim modulima)

- mora biti fleksibilna u smislu da dozvoli specifikaciju različitih tipova komunikacije između korisnika i programa (sistem menija, sistem "odgovori-na-pitanje", sistem "popuni formular", sistem komandnih tipki, ...)

Osim navedenih zahtjeva, poželjno je da se specifikacija modula, urađena danom metodologijom, što je moguće lakše realizira u odgovarajućem programskom jeziku.

Za specifikaciju gramatike jezika računa predikata prvog reda pogodna je BN-specifikacija jer među alatima PDC Prologa postoji program za generiranje sintaktičkog analizatora za gramatike koje su na taj način specificirane. Tako generirani sintaktički analizator predstavlja jedan od modula (predprocesora) deduktivnog mehanizma.

Za specifikaciju korisničkog sučelja pogodna je CSP-specifikacija jer se preko nje može opisati kvaziparalelnost komunikacije između aplikativnog programa i korisnika, dok je SF-specifikacija pogodna za specifikaciju funkcija editora jer se njegove operacije mogu opisati kao preslikavanja među nizovima znakova radne abecede. U narednom odjeljku dajem BN - specifikaciju gramatike jezika računa predikata prvog reda, dok je odjeljak 4.4. posvećen SF - specifikaciji fragmenta (funkcija) hipotetičkog editora, od kojih će neke poslužiti u odjeljku 5.3 za ilustraciju mogućnosti PDC Prologa kao jezika za implementaciju

4.3. BN - specifikacija gramatike jezika formula računa predikata prvog reda

Da bi deduktivni mehanizam bio u stanju riješiti problem, skup formula kojim je problem opisan mora biti na odgovarajući način preoblikovan. Kako, to ovisi o konkretnom deduktivnom mehanizmu. Primjer preoblikovanja skupa formula kojim je problem opisan, za slučaj deduktivnog mehanizma zasnovanog na klasičnom pravilu rezolucije za račun predikata, dan je u odjeljku 3.2. Svakako, elementarno svojstvo koje se zahtijeva od svake formule je da ona bude građena prema pravilima sintakse dotičnog jezika, u ovom slučaju računa predikata prvog reda. Postoji više međusobno ekvivalentnih načina specifikacije građe formula (općenito, gramatike formalnog jezika). Među njima izdvajamo specifikaciju konačnim automatima i specifikaciju tipa Bacusa-Naura (BN-specifikacija), poznatu iz domene specifikacije programskih jezika.

Ovdje dajemo primjer BN-specifikacije za gramatiku jezika računa

predikata prvog reda. Prije same specifikacije dajemo opis abecede jezika i opis sredstava specifikacije, na način primjeren ograničenjima vezanim za implementaciju.

Abeceda jezika računa predikata prvog reda je unija slijedećih skupova:

$\mathcal{A}_1 = \{0, 1, 2, \ldots, 9\}$ -osnovni skup indeksa
$\mathcal{A}_2 = \{x, y, z, u, v, w\}$ -osnovni skup varijabli
$\mathcal{A}_3 = \{c, d, e\}$ -osnovni skup konstanti
$\mathcal{A}_4 = \{f, g, h\}$ -osnovni skup funkcijskih znakova (funkcija)
$\mathcal{A}_5 = \{P, Q, R, S\}$ -osnovni skup predikatnih znakova (predikata)
$\mathcal{A}_6 = \{ \neg, \vee, \wedge, \rightarrow, \leftrightarrow \}$ -skup logičkih veznika
$\mathcal{A}_7 = \{ \forall, \exists \}$ -skup kvantifikatora "za svaki" i "postoji"
$\mathcal{A}_8 = \{(,), ,\}$ -zagrade i zarez

BN-specifikaciju čini skup njenih produkcija (pravila gradnje) . Osnovni oblik produkcije BN-specifikacije je:

<ime–neterminala> := "niz neterminala i (ili) terminala"
Znak ":=" može se čitati kao "ima strukturu" ili "istovjetan je"
Elemente relacije := zovemo konstruktima.
Konstrukti se dijele na terminalne i neterminalne. Među neterminalnim konstruktima jedan je startni. On predstavlja "finalni proizvod" jezika, u ovom slučaju formulu jezika računa predikata kao apstraktni pojam. Ostali neterminali predstavljaju "međuproizvode" jezika. Terminalni konstrukti mogu se pojavljivati samo na desnim stranama produkcija.

U zapisima produkcija koristi se i nekoliko pomoćnih znakova.

"*" u zapisu <konstrukt>* znači da se <konstrukt> (i (ili) njegove izvedenice) nižu nula ili više puta, dok zapis <konstrukt>+ kazuje da je to slučaj bar jedanput.

Zapis [<neterminal>] znači da se <neterminal> u produkciji pojavljuje najviše jedanput.

Znak "|" koristi se u značenju veznika "ili".

<formula> := (<formula>)| <unarni-veznik><formula> |
| <formula> <binarni-veznik><formula>|
| <kvantifikator><varijabla> <formula> | <atomarna-formula>

<unarni-veznik> := ¬
<binarni-veznik> := ∨ | ∧ | → | ↔
<kvantifikator> := ∀ | ∃
<atomarna-formula> := <predikat>(<term><zarez-term>*)
<predikat> := <ime-predikata>[<indeks>]
<ime-predikata> := P | Q | R | S
<indeks> := <znamenka>+
<znamenka> := 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<zarez-term> := ,<term>
<term> := <funkcija>(<argument><zarez-argument>*) | <argument>
<funkcija> := <ime-funkcije>[<indeks>]
<ime-funkcije> := f | g | h
<zarez-argument> := ,<argument>
<argument> := <konstanta> | <varijabla> | <term>
<konstanta> := <ime-konstante><indeks>
<ime-konstante> := c | d | e
<varijabla> := <ime-varijable><indeks>
<ime-varijable> := x | y | z | u | v | w

Startni konstrukt ovdje je konstrukt <formula> u prvoj produkciji. Tom produkcijom ujedno se zadaje pravilno i konzistentno rukovanje zagradama.

4.4. Primjer formalne specifikacije fragmenta grafičkog editora

Ulaz u editor predstavlja neformatirani tekst (neprekinuti niz znakova) u abecedi koja je unija skupa znakova od kojih se grade riječi (na primjer engleski alfabet) i dvaju specijalnih znakova: *bjelina* ("prazni znak", ASCII 13) i *prijelom* (ASCII 32). Zbog daljnjih referenci uvodimo oznake *AB* i *AL* za abecedu odnosno alfabet te *PZ* (posebni znakovi) za skup { bjelina, prijelom}. Očito vrijedi jednakosti $AB = AL \cup PZ$. Ulazni tekst prelomljen je u konačan broj redaka znakom *prijelom*.

Izlaz iz editora mora biti formatirani tekst podijeljen u retke jednake (unaprijed zadane) duljine takav da poredak riječi bude kao u ulaznom tekstu i da ispred prve riječi izlaznog teksta nema bjelina.

Da bi se formatiranje teksta moglo realizirati na traženi način,

nužno je imati predikat kojim se riječi definiraju kao objekti, predikat koji daje redni broj riječi u tekstu i predikat koji broji ukupan broj riječi u tekstu. Konstrukte jezika SF-specifikacije prema [Berztiss, 1990] za prva dva predikata i potrebne komentare dajem u "hodu". U odjeljku 5.3 predikat riječ(Tekst, i, j) implementiran je u PDC Prologu.

riječ(Tekst:String, i, j: 1...duljina(Tekst))≡
$i \leq j$ (1)
$\wedge$ $(i \neq 1) \rightarrow$ element(Tekst(i - 1), *PZ*) (2)
$\wedge$ $(j \neq$ duljina(Tekst)) $\rightarrow$ element(Teks(i+1), *PZ*) (3)
$\wedge$ SveOperacije($\wedge$:(element(Tekst(k), *AL*)| $i \leq k \leq j$) (4)

Riječima: riječ je objekt tipa String koji počinje njegovim i-tim znakom i završava j-tim znakom ako:
prvi znak riječi prethodi posljednjem (uvjet 1)
i ako početni znak riječi nije prvi znak teksta, onda je prethodni znak teksta posebni znak (uvjet 2)
i ako završni znak riječi nije posljednji znak teksta, onda je naredni znak teksta posebni znak (uvjet 3)
i svi znakovi riječi su znakovi alfabeta (uvjet 4)

k-taPoReduRiječ(Tekst:String, i, j, k: RedniBrojevi)≡
riječ(Tekst, i, j)
$\wedge$ (k=1) $\rightarrow$ SveOperacije($\wedge$: (element(Tekst(t), *PZ*)| $1 \leq t < i$)
$\wedge$ (k>1) $\rightarrow$ SveOperacije($\vee$: (k-taPoReduRiječ(Tekst, k-1, u, v)$\wedge$
$\wedge$ SveOperacije($\wedge$: (element(Tekst(t), *PZ*)| $v < t < i$, u, v : RedniBrojevi)

Riječima: Riječ riječ(Tekst, i, j) je k-ta riječ teksta Tekst ako:
kad je to prva riječ teksta, onda su svi znakovi koji joj prethode posebni znakovi
i kad nije prva riječ i kad počinje u-tim znakom teksta i završava v-tim znakom, onda su svi znakovi od (v+1)-og do (i-1)-og zaključno, posebni.

Među predikatima koji opisuju izlazni tekst je predikat prijelomJeDobar čiji opis riječima dajem nakon specifikacije:

prijelomJeDobar(PrelomljeniTekst:String)≡
ne(element(PrelomljeniTekst(1), *PZ*))∧
∧ ne(element(PrelomljeniTekst(duljina(PrelomljeniTekst)), *PZ*) ∧
∧ SveOperacije(∧(element(PrelomljeniTekst(j), *PZ*) →
→ne(element(PrelomljeniTekst(j+1), *PZ*) | 1 ≤ j < duljina(PrelomljeniTekst)))

Riječima: Izlazni tekst je dobro prelomljen ako:
njegov prvi znak nije poseban znak
i njegov posljednji znak nije poseban znak
i ako je neki njegov znak poseban znak, onda takav ne može biti i naredni znak teksta

5. Jezik Prolog kao razvojna sredina za implementaciju ljuske ekspertnih sistema

5.1. PDC Prolog

Standardom jezika Prolog smatra se njegov opis dan u knjizi [Clocksin & Mellish,1984], poznat i kao C & M prolog. Ovdje smatramo taj standard poznatim i o njemu neće posebno biti govora. Pojedine implementacije u okviru klasične Von Neumannove arhitekture računala razlikuju se po vrstama i broju u jezik ugrađenih (built in) predikata. Predikate tog tipa zvat ćemo i primitivnim predikatima ili primitivima. Primitivi dalje služe kao gradivni elementi za predikate koje definira korisnik (programer).

PDC Prolog je razvijen u firmi Prolog Development Center sa sjedištima u Danskoj i SAD. Postoje verzije za rad pod operativnim sistemima MS DOS, OS/2 i UNIX. Jezik je razvijen na osnovi jezika Turbo Prolog firme Borland. Ujedno je i interpreter i prevodilac. Devet desetina njegovog koda napisano je u jeziku C i ostatak u assembleru. Ovdje navodimo neka globalna svojstva jezika:

— PDC Prolog je brz, visoko optimiziran prevodilac koji se po brzini može mjeriti s prevodiocima za C i Pascal

— ima ugrađen sistem za otkrivanje grešaka tokom procesa prevođenja programa

— ima integriranu okolinu za razvoj izvršnih verzija programa bez faze povezivanja vanjskim programima za povezivanje pojedinih modula (linkerima)

— ima ugrađen sistem za rukovanje vanjskim bazama podataka sa preko pedeset primitiva za rukovanje s B^+ stablima, proširenom RAM memorijom EMS tipa i višekorisničkom pristupu istim datotekama

— ima ugrađen editor koji se može proširiti vanjskim (korisnikovim) naredbama i koji se može ugraditi u korisničke programe

— ima ugrađen sistem za izvještavanje o greškama s mogućnošću odabira nivoa detaljizacije od strane korisnika

— ima ugrađeno sučelje za povezivanje s drugim jezicima čiji prevodioci generiraju standardne objektne (.OBJ) datoteke

— podržava video-izlaz visoke rezolucije

— ima preko šezdeset primitiva za podršku grafici

5.2. Klasifikacija primitiva i struktura programa jezika PDC Prolog

Da bi se mogli razumjeti programi koji slijede, ovdje ukratko dajem grubu klasifikaciju primitiva jezika PDC Prolog te opisujem globalnu strukturu programa u tom jeziku u nužnom obimu.

Primitive jezika PDC Prolog moguće je prirodno razvrstati u nekoliko osnovnih klasa, i to:

- primitive za upravljanje datotekama
- primitive za procesiranje nizova znakova (stringova)
- primitive za rad s internim bazama podataka
- primitive za rad s eksternim bazama podataka
- primitive za rad s listama
- primitive za rad s ekranom i prozorima
- primitive za rad s grafikom
- primitive ! (rez), *fail* (neuspjeh), *getbacktrack* i *cutbacktrack* za statičko i dinamičko upravljanje rezom i forsiranje potrage za novim rješenjima problema
- primitive za trasiranje izvršenja programa
- primitive - relacije uređaja, aritmetičke i logičke operacije,...

Globalna struktura programa je slijedeća:

```
constants
    /* sekcija u kojoj se deklariraju konstante, npr. pi=3,14 */
domains
    /* sekcija u kojoj se deklariraju domene predikata ako to
nisu neke od sistemskih domena (integer, real, char, string i symbol) */
predicates
    /* sekcija u kojoj se deklariraju predikati poput predikata
riječ(Tekst,i,j) iz odjeljka 5. 2 */
clauses
    /* sekcija u kojoj se predikati definiraju do nivoa primitiva */
goal
    */ sekcija u kojoj se definira cilj programa čije dostizanje
rješava postavljeni problem */
```

5.3. Implementacija fragmenta specifikacije editora u PDC Prologu

```
/*****************************************************************/
/* PDC Prolog program koji u zadanom stringu provjerava da li je */
/* zadani podstring rijec u zadanoj abecedi. Podstring se zadaje */
/* indeksima svojih krajnjih znakova.                            */
/*****************************************************************/
/*****************************************************************/
/* Predikat rijec(tekst,i,j) je jedan od predikata u specifikaciji */
/* i implementaciji operacije prijeloma retka (uz zanemarivanje    */
/* suvisnih bjelina i uz izravnavanje duljine retka na zadanu      */
/* duljinu) u okviru specifikacije editora.                        */
/*****************************************************************/

domains

    i, j, vrijednost = integer
    tekst, podtekst = string
    znak = char
    lista=integer*
```

```
predicates

        element(vrijednost, lista)
        rijec(tekst, i, j)
        znakIspredJePoseban(tekst, i)
        znakIzaJePoseban(tekst, j)
        podTekst(tekst, i, j, podtekst)
        znakoviPodtekstaSuObicni(podtekst)
        posaljiPoruku(tekst, I, J)

clauses

        element(H,[H|_]).
        element(H,[_|T]):-element(H,T).

        rijec(Tekst,I,J):-
                I<=J,
                znakIspredJePoseban(Tekst, I),
                znakIzaJePoseban(Tekst,J),
                podTekst(Tekst, I, J, PodTekst),
                znakoviPodTekstaSuObicni(PodTekst).

        znakIspredJePoseban(Tekst,1).

        znakIspredJePoseban(Tekst,I):-
                I<>1, I1 = I-1,
                subchar(Tekst,I1,Znak),
                char_int(Znak,Vrijednost),
                element(Vrijednost,[13,32]).

   znakIzaJePoseban(Tekst,J):-
                str_len(Tekst, Duljina),
                J <> Duljina,
                J1 = J+1,
                subchar(Tekst, J1, Znak),
                char_int(Znak,Vrijednost),
                element(Vrijednost,[13,32]).
```

```
        znakIzaJePoseban(Tekst,J):-
                str_len(Tekst, Duljina),
                J = Duljina,!.

        podTekst(Tekst, I, J, PodTekst):-
                J1 = J - I + 1,
                substring(Tekst, I, J1, PodTekst).

        znakoviPodTekstaSuObicni(PodTekst):-
                frontchar(PodTekst, Znak, PodTekst1),
                char_int(Znak, Vrijednost),
                Vrijednost<>13,
                Vrijednost<>32,
        znakoviPodTekstaSuObicni(PodTekst1).

        znakoviPodtekstaSuObicni(Podtekst):-
                str_len(PodTekst, Duljina), Duljina = 0, !.

        posaljiPoruku(Tekst, I, J):-
                rijec(Tekst, I, J),
                write(" Tekst predstavlja pravu rijec! "), !.

        posaljiPoruku(_,_,_):-write(" Tekst NE predstavlja pravu rijec! ").

goal

    makewindow(1,112, 25,"Demonstracija implementacije specifikacije",
                2,2,20,75),
    cursor(1,2),
    write("Upisi tekst"),
    cursor(2,2),
    readln(Tekst),
    cursor(Redak,_),
    Redak1=Redak + 1,
    cursor(Redak1,2),
    write("Indeks prvog znaka u podstringu je"),
    readint(I),
```

```
cursor(Redak2,—),
Redak3 = Redak2 +1,
cursor(Redak3, 2),
write("Indeks zadnjeg znaka u podstringu je"),
readint(J),
Redak4 = Redak3 +1,
cursor(Redak4,2),
posaljiPoruku(Tekst, I, J).
```

Literatura:

[Alexander,1990]: H. Alexander: *Structuring dialogues using CSP,* u zborniku M. Harrison, H. Thimbleby (eds.): *Formal methods in human-computer interaction,* Cambridge University Press, 1990.

[Berztiss, 1990]: A. Berztiss: *Formal specification methods and visualization,* u zborniku Shi – Kuo Chang (ed.): Principles of visual programming systems, Prentice- Hall, 1990.

[Bhatta & Karnick, 1991]: K.S.H.S.R. Bhatta, H. Karnick: *A resolution rule for well formed formule,* Theoretical Computer Science, vol. 81, (2),1991, str. 223 – 236.

[Bizam & Herczeg, 1972]:D. Bizam, J. Herczeg: *JATEK ES LOGIKA, 85 FELADATBAN,* ruski prijevod, "MIR", Moskva, 1973.

[Bizam & Herczeg, 1975]:D. Bizam, J. Herczeg: *SOKSZINU LOGIKAI, (175 FELADAT),* ruski prijevod, "MIR", Moskva, 1978.

[Clocksin & Mellish, 1984]: W.F. Clocksin, C.S. Mellish: *Programming in Prolog,* Springer-Verlag, 1984.

[Čubrilo, 1989]: M. Čubrilo: *Matematička logika za ekspertne sisteme,* "Informator", Zagreb, 1989.

[Hashim, 1990]: S. H. Hashim: *Exploring hypertext programming: Writing knowledge representation and problem-solving programs,* Windcrest Books, 1990.

[Jaglom, 1968]: I. M. Jaglom: *Kak razrjezat kvadrat?*, "Nauka", Moskva, 1968.

[Maier, 1983]: D. Maier: *The Theory of Relational Databases,* Computer Science Press, 1983.

[Stiny, 1975]: G. Stiny: *Pictorial and formal aspects of shape and shape grammars,* Birkhauser, 1975.

[Sufrin, 1982]: B. Sufrin: *Formal specification of a display - oriented editor,* Science of Computer Programming 1 (1982) 157 - 202.

Primljeno: 1991 – 10 – 02

Čubrilo M. Some Aspects of Expert Systems Shell Modelling

Summary

This article is concerned with some aspects of expert systems shell modelling. It primarily relates to the formal methods of specification of the modules of a shell. In connection with the supposed problem domains, it is of special interest to develop a graphical editor for formulae from the language of the first order predicate calculus.

Here, we propose BN-specification of the grammar of that language. In the sequel, we analyse a sample of SF-specification of a fragment of a hypotethical editor and demonstrate the possibility of implementation of some of its functions in the PDC Prolog programming environment.